Contenido

1 Introducción

Los programas de computador comerciales para el análisis y diseño de estructuras que se encuentran vigentes a la fecha cuentan, en general, con un entorno gráfico que le permite al usuario describir el modelo de forma interactiva, procesarlo y visualizar los resultados de manera conveniente.

En Escamilla, 1995 se presenta una lista de algunos de estos programas de uso común en América Latina, entre los cuales se encuentra *ETABS* (Three Dimensional Analysis of Building Systems - Extended Version).

ETABS es un programa de computador creado por Edward Wilson, Jeffery Hollings y Henry Dovey en 1975. Según Wilson et al, 1975, este programa de computador fue desarrollado para el análisis estructural lineal de edificios de pórticos y muros a cortante sujetos tanto a cargas estáticas como sísmicas. El edificio es idealizado como un sistema de elementos tipo pórticos y muros a cortante independientes interconectado por losas de entrepiso las cuales son rígidas en su propio plano.

Este programa es una extensión de *TABS* (Three Dimensional Analysis of Building Systems) para poder analizar pórticos en tres dimensiones. Según Wilson y Dovey, 1972, una de las razones para desarrollar TABS fue darle una retroalimentación a los usuarios de los programas *FRMSTC* (Static Load Analysis of High-Rise Buildings), *FRMDYN* (Dynamic Analysis of Multistory Buildings), *LATERAL* y *SOLID SAP* (Static Analysis Program for Three-Dimensional Solid Structures).

FRMSTC permitía analizar edificios simétricos con pórticos y muros a cortante paralelos sujetos a cargas estáticas y evaluar los modos y las frecuencias. FRMDYN era similar a FRMSTC con la excepción que la carga era la aceleración del terreno debido a un desplazamiento dependiente del tiempo. LATERAL fue una extensión de FRMSTC que permitía analizar linealmente pórticos y muros a cortante que no eran necesariamente paralelos con tres grados de libertad en cada piso. SOLID SAP era un programa general de elementos finitos y tenía una opción que permitía introducir la aproximación de piso rígido. Este programa también tenía la opción de realizar análisis dinámico.

En la actualidad, ETABS se encuentra en la versión 18.1.1 y según Computers y Structures,

2020, puede ser ejecutado en computadores con sistema operativo Windows 7, Windows 8 o Windows 10 con arquitectura de 64 bits que cuenten como mínimo con un procesador Intel Pentium 4 o AMD Athlon 64, una resolución de 1024x768 pixeles con 16 bits por canal, 8 GB de RAM y 6 GB de espacio en el disco duro. En la figura **1-1** se presenta la ventana del programa ETABS ejecutandose en un computador con Windows 10.

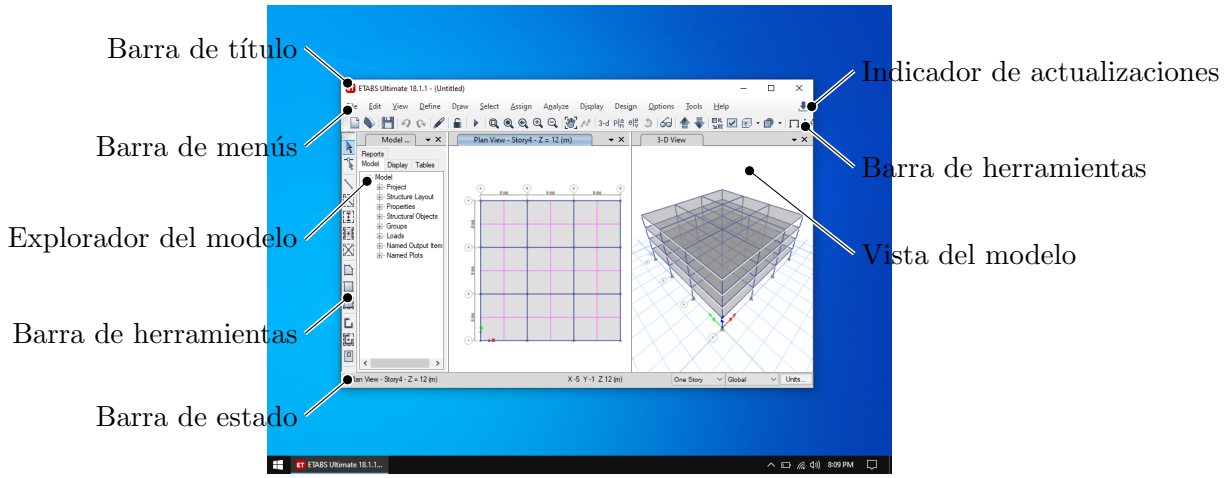

Figura 1-1: Ventana del programa ETABS ejecutandose en Windows 10.

A través de múltiples cuadros de diálogo, los cuales son accesibles ya sea a través de la barra de menús, las barras de herramientas, el explorador del modelo, las vistas del modelo o con atajos de teclado, el usuario es capaz de modelar la estructura que desea analizar al describir los materiales, las secciones transversales, los elementos estructurales, las condiciones de apoyo, los diafragmas y las cargas.

Según Computers y Structures, 2017, ETABS analiza el modelo usando el motor de análisis *SAPFire*, el cual es común a otros programas de la misma compañia (*SAP2000*, *SAFE* y *CSiBridge*). SAPFire es la última versión de la serie de programas *SAP* y ofrece las siguientes herramientas:

- Análisis estático y dinámico,
- Análisis lineal y no lineal,
- Análisis sísmico y análisis incremental no lineal (*pushover*),
- Análisis de cargas móviles,
- No linealidad geométrica, incluyendo efectos P-delta y grandes desplazamientos,
- Etapas constructivas,

- Fluencia lenta (*creep*), retracción (*shrinkage*) y envejecimiento,
- Análisis de pandeo,
- Análisis de densidad espectral de potencia y estado estacionario,
- Elementos tipo pórtico y laminares, incluyendo el comportamiento de vigas, columnas, cerchas, membranas y placas,
- Elementos tipo cable y tendón,
- Elementos bidimensionales planos y elementos sólidos asimétricos,
- Elementos sólidos tridimensionales,
- Resortes no lineales y apoyos,
- Propiedades de los resortes y apoyos dependientes de la frecuencia,

Con los resultados del análisis del modelo, el posprocesador de ETABS puede *diseñar* los elementos estructurales de acuerdo a uno de varios códigos de diseño de diferentes países. ETABS es capaz de diseñar pórticos en acero, pórticos en concreto, vigas compuestas, columnas compuestas, vigas en acero de alma abierta (*steel joist*), muros a cortpante y losas de concreto.

Adicionalmente, según Computers y Structures, 2019, ETABS cuenta con la posibilidad de generar dibujos estructurales esquemáticos de las plantas estructurales, de los despieces de vigas, columnas y muros a cortante, y de los detalles de las conexiones de acero.

En términos generales, estos programas de computador comerciales cuenta con características similares a las de ETABS. Actualmente, dichos programas están innovando para permitirle al usuario trabajar con modelos *BIM* (Building Information Modeling).

1.1. Objetivo

Desarrollar un programa de computador a código abierto para el análisis de estructuras tridimensionales tipo pórtico sometidas a cargas estáticas.

Con este trabajo se pretende contribuir al ejercicio libre de la profesión del ingeniero estructural y a la enseñanza del análisis de las estructuras.

1.1.1. Objetivos específicos

- Desarrollar el ambiente gráfico y la interfaz gráfica de usuario del programa de computador para permitirle al usuario ingresar los datos que describen la estructura, las acciones a las cuales se encuentra sometida y visualizar los resultados del análisis estructural.

- Desarrollar el módulo de análisis estructural para calcular el desplazamiento de los nudos, el valor de las reacciones y de las fuerzas internas de los elementos de una estructura sometida a cargas estáticas.

1.2. Metodología

Se desarrollaron los programas de computador *pyFEM* y *FEM.js*. El primero para analizar estructuras tridimensionales tipo pórtico sometidas a cargas estáticas y el segundo para modelarlas. Esto con el fin que FEM.js pueda ser usado junto con otro programa de computador diferente a pyFEM.

Durante el desarrollo de estos programas se utilizó *git* como sistema de control de versiones. Según Chacon, 2014, git es un sistema distribuido de control de versiones que registra los cambios realizados a un conjunto de archivos para coordinar el trabajo entre programadores.

Una copia de los repositorios de pyFEM y FEM.js se encuentran en la página de internet *GitHub*, la cual permite alojar proyectos utilizando git. pyFEM está alojado en https://github.com/rvcristiand/pyFEM mientras que FEM.js está alojado en https://github.com/rvcristiand/FEM.js.

1.2.1. pyFEM

pyFEM fue desarrollado en *Python*. Según Lutz, 2013, Python es un lenguaje de programación interpretado orientado a objetos cuya filosofía hace enfasis en la legibilidad de su código. Los archivos revelantes que componen el repositorio de pyFEM son:

```
pyFEM/
├── LICENSE
├── README.md
├── example_1.json
├── example_2.json
├── example_3.json
├── pyFEM/
│   ├── classtools.py
│   ├── core.py
│   └── primitives.py
└── test/
    ├── space_frame.py
    └── trusses.py
```

El archivo `LICENCE` contiene la licencia de pyFEM, la cual se presenta a continuación.

> MIT License
>
> Copyright (c) 2019 pyFEM
>
> Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
>
> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
>
> THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

El archivo `README.md` contiene todas las instrucciones necesarias para ejecutar y usar pyFEM.

Los archivos `example_1.json`, `example_2.json` y `example_3.json` almacenan los modelos de tres de los ejemplos presentados en Escamilla, 1995 que han sido analizados con pyFEM. La extensión *json* se usa para indicar que los archivos tienen formato *JSON* (de sus siglas en inglés JavaScript Object Notation), el cual es un formato sencillo para el intercambio de datos. El modelo es descrito de tal manera que puede ser interpretado por FEM.js para generar su representación en una escena tridimensional.

La carpeta *pyFEM* contiene los archivos `classtools.py`, `core.py` y `primitives.py` los cuales contienen las instrucciones para analizar los modelos. La extensión *py* se usa para indicar que los archivos son programas de Python.

En el archivo `classtools.py` se encuentran las *clases* `UniqueInstances` y `AttrDisplay`, la primera para evitar que se creen *objetos* de una misma clase con la misma información y la segunda para generar una representación conveniente de los objetos.

En el archivo `primitives.py` se encuentran varias clases, entre ellas `Material`, `Section`, `Joint`, `Frame`, `Support`, `LoadPattern`, etc., las cuales permiten describir los diferentes atributos del modelo.

En el archivo `core.py` se encuentra la clase `Structure` la cual permite describir estructuras para ser analizados. Para crear objetos de esta clase se debe llamar la clase indicando los grados de libertad a tener en cuenta. A partir de un objeto de esta clase es posible describir el modelo de la estructura al agregar materiales, secciones transversales, nodos, elemenetos tipo pórtico, apoyos, patrones de carga, cargas en los nodos y cargas distribuidas en los elementos tipo pórtico.

En el ejemplo 1.2.1 se presenta la solución a un ejercicio de Escamilla, 1995 usando pyFEM.

Ejemplo

Resuelva completamente la cercha mostrada por el método matricial de los desplazamientos. El material es acero estructural con $E = 2040\ t/cm^2$. Las áreas están dadas entre paréntesis en cm^2.

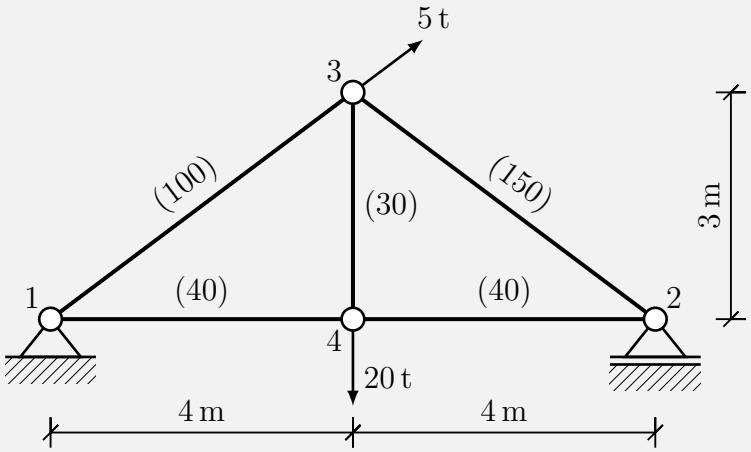

Figura 1-2: Cercha simple plana del *Ejemplo 7.1* de Escamilla, 1995.

Solución **-** En el algoritmo 1.1 se presenta un programa de Python para solucionar el modelo de la estructura usando pyFEM. Las instrucciones consisten en crear un nuevo objeto tipo `Structure`, al cual se le ha dado el nombre de *structure* y, seguidamente, se agregan los materiales, las secciones, los nodos, los elementos tipo cercha, los apoyos, los patrones de carga y las cargas en los nodos. Una vez los datos del modelo de la estructura han sido ingresados, se puede solucionar el modelo mediante la instrucción `structure.solve()`.

Algoritmo 1.1: Ingreso de los datos del modelo de la estructura a *pyFEM*.

```
# create the model
model = Structure(ux=True, uy=True)

# add materials
model.add_material(key='1', modulus_elasticity=2040e4)

# add sections
model.add_section(key='1', area=030e-4)
model.add_section('2', area=040e-4)
model.add_section('3', area=100e-4)
model.add_section('4', area=150e-4)

# add joints
```

```
model.add_joint(key=1, x=0, y=0)
model.add_joint(2, 8, 0)
model.add_joint(3, 4, 3)
model.add_joint(4, 4, 0)

# add frames
model.add_frame(key='1-3', key_joint_j=1, key_joint_k=3, key_material='1'
    , key_section='3')
model.add_frame('1-4', 1, 4, '1', '2')
model.add_frame('3-2', 3, 2, '1', '4')
model.add_frame('4-2', 4, 2, '1', '2')
model.add_frame('4-3', 4, 3, '1', '1')

# add supports
model.add_support(key_joint=1, ux=True, uy=True)
model.add_support(2, ux=False, uy=True)

# add load patterns
model.add_load_pattern(key='point loads')

# add point loads
model.add_load_at_joint(key_load_pattern='point loads', key_joint=3, fx=5
    * 0.8, fy=5 * 0.6)
model.add_load_at_joint('point loads', 4, fy=-20)

# solve the problem
model.solve()

print(model)

# export the model
model.export('example_1.json')
```

Cuando se ejecuta la instrucción `structure.solve()`, `pyFEM` comienza a solucionar el modelo de la estructura con base en la información ingresada. Los pasos que se efectúan para solucionar la estructura consisten en: (1) asignar los grados de libertad de los nodos, (2) ensamblar la matriz de rigidez del modelo de la estructura, (3) imponer las condiciones de apoyo en la matriz de rigidez del modelo, (4) ensamblar el vector de fuerzas en los nodos para cada uno de los patrones de carga, (5) imponer las condiciones de apoyo en el vector de fuerzas en los nodos para cada caso de carga, (6) encontrar los desplazamientos de los nodos para cada patrón de carga, (7) encontrar las reacciones en los apoyos para cada patrón de carga y (8) guardar la solución en los nodos y en los apoyos para cada patrón de carga.

A continuación se presenta el resultado de cada uno de los pasos realizados por `pyFEM` para

solucionar la cercha del ejemplo 7.1.

Grados de libertad de los nodos

Para realizar el ensamblaje de la matriz de rigidez del modelo de la estructura y del vector de fuerzas de los nodos, se asignan números a los grados de libertad de los nodos de la estructura en el orden en que fueron ingresados.

Con base en ésto y en el algoritmo 1.1, el nodo *'1'* se le han asignado los grados de libertad *0, 1* y *2*, al nodo *'2'* los grados de libertad *3, 4* y *5*, y así sucesivamente.

Matrices de rígidez

Una vez se establecen los grados de libertad de los nodos del modelo de la estructura, se ensambla la matriz de rigidez del modelo de la estructura. Este proceso consiste en calcular una a una las matrices de rigidez de los elementos ensamblandolas en la matriz de rigidez del modelo, la cual, inicialmente, es una matriz de ceros.

Aunque no se almacenan las matrices de rigidez de cada uno de los elementos del modelo de la estructura, el usuario puede consultarlas. En la tabla **1-1** se presenta la matriz de rigidez en coordenadas globales del elemento *1-3*, con sus respectivos grados de libertad, la cual se obtiene mediante la instrucción *structure.trusses['1-3'].get_global_stiff_matrix()*.

	0	*1*	*2*	*6*	*7*	*8*
0	26112	19584	0	-261112	-19584	0
1	19584	14688	0	-19584	-14688	0
2	0	0	0	0	0	0
3	-26112	-19584	0	26112	19584	0
4	-19584	-14688	0	19584	14688	0
5	0	0	0	0	0	0

t/m

Tabla 1-1: Matriz de rigidez en coordenadas globales del elemento *1-3*.

En las tablas **1-2** a la **1-5** se presentan las matrices de rigidez en coordenadas globales de los demás elementos de la estructura, las cuales se obtienen con instrucciones similares a la anterior.

	0	*1*	*2*	*9*	*10*	*11*	
0	20400	0	0	-20400	0	0	t/m
1	0	0	0	0	0	0	
2	0	0	0	0	0	0	
9	-20400	0	0	20400	0	0	
10	0	0	0	0	0	0	
11	0	0	0	0	0	0	

Tabla 1-2: Matriz de rigidez en coordenadas globales del elemento *1-4*.

	6	*7*	*8*	*3*	*4*	*5*	
6	39168	-29376	0	-39168	29376	0	t/m
7	-29376	22032	0	29376	-22032	0	
8	0	0	0	0	0	0	
3	-39168	29376	0	39168	-29376	0	
4	29376	-22032	0	-29376	22032	0	
5	0	0	0	0	0	0	

Tabla 1-3: Matriz de rigidez en coordenadas globales del elemento *3-2*.

	9	*10*	*11*	*6*	*7*	*8*	
9	0	0	0	0	0	0	t/m
10	0	20400	0	0	-20400	0	
11	0	0	0	0	0	0	
6	0	0	0	0	0	0	
7	0	-20400	0	0	20400	0	
8	0	0	0	0	0	0	

Tabla 1-5: Matriz de rigidez en coordenadas globales del elemento *4-3*.

Al comparar los resultados obtenidos con las matrices de rigidez de cada uno de los elementos presentados en Escamilla, 1995 se encuentra que se trata de los mismos valores.

Así mismo como el usuario puede indagar por la matriz de rigidez en coordenadas globales de cada elemento, puede hacerlo para el modelo de la estructura, mediante la instrucción *structure.get_k()*. Al hacerlo, obtiene los valores presentados en la tabla **1-6**.

Una vez más, al comparar los resultados obtenidos con la matriz de rigidez del modelo de la estructura presentado en Escamilla, 1995 se encuentra que se trata de los mismos valores.

	9	*10*	*11*	*3*	*4*	*5*
9	20400	0	0	-20400	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
3	-20400	0	0	20400	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

t/m

Tabla 1-4: Matriz de rigidez en coordenadas globales del elemento *4-2*.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
0	46512	19584	0	0	0	0	-26112	-19584	0	-20400	0	0
1	19584	14688	0	0	0	0	-19584	-14688	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	59568	-29376	0	-39168	29376	0	-20400	0	0
4	0	0	0	-29376	22032	0	29376	-22032	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0
6	-26112	-19584	0	-39168	29376	0	65280	-9792	0	0	0	0
7	-19584	-14688	0	29376	-22032	0	-9792	57120	0	0	-20400	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	-20400	0	0	-20400	0	0	0	0	0	40800	0	0
10	0	0	0	0	0	0	0	-20400	0	0	20400	0
11	0	0	0	0	0	0	0	0	0	0	0	0

t/m

Tabla 1-6: Matriz de rigidez en coordenadas globales del modelo de la estructura.

Sin embargo, se debe reordenar una de las dos matrices para encontrar los valores en las mismas posiciones de la matriz.

Obtenida la matriz de rigidez de la estructura, se procede a imponer las condiciones de apoyo del modelo de la estructura. Esto se realiza modificando la estructura, tal como se mostró en el capítulo **??**. En la tabla **1-7** se presenta dicho resultado.

Vector de fuerzas

Así como se deben encontrar las matrices de rigidez de cada uno de los elementos del modelo de la estructura para posteriormente *ensamblarlas*, se deben encontrar las acciones en los nodos de los elementos para cada patrón de carga.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
0	1	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	0	0	0	
2	0	0	1	0	0	0	0	0	0	0	0	0	
3	0	0	0	59568	0	0	-39168	29376	0	-20400	0	0	
4	0	0	0	0	1	0	0	0	0	0	0	0	
5	0	0	0	0	0	1	0	0	0	0	0	0	
6	0	0	0	-39168	0	0	65280	-9792	0	0	0	0	t/m
7	0	0	0	29376	0	0	-9792	57120	0	0	-20400	0	
8	0	0	0	0	0	0	0	0	1	0	0	0	
9	0	0	0	-20400	0	0	0	0	0	40800	0	0	
10	0	0	0	0	0	0	0	-20400	0	0	20400	0	
11	0	0	0	0	0	0	0	0	0	0	0	1	

Tabla 1-7: Matriz de rigidez en coordenadas globales del modelo de la estructura con las condiciones de apoyo impuestas.

El usuario puede indagar por dicho vector, para el patrón de carga *point loads* mediante la instrucción `structure.load_patterns['point loads'].get_f()`. Con dicha instrucción, el usuario obtiene los datos que se muestran en la tabla **1-8**.

Obtenido el vector de fuerzas para dicho patrón de carga, se imponen las condiciones de apoyo del modelo de la estructura. Debido a que los desplazamiento en los apoyos son iguales a cero, el vector de fuerzas en los nodos no varia.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
0	0	0	0	0	0	4	3	0	0	-20	0	^{t} t

Tabla 1-8: Vector de fuerzas de los nodos del modelo de la estructura para el patrón de carga *point loads*.

Vector de desplazamientos

Al imponerse las condiciones de apoyo a la matriz de rigidez del modelo de la estructura, se determinan los desplazamientos de los nodos de la estructura para cada uno de los patrones de carga, donde se ha calculado el vector de fueras en los nodos de cada patrón de cargas y se le han impuesto las condiciones de apoyo.

En la tabla **1-9** se presenta el vector de desplazamientos en los nodos del modelo de la estructura para el patrón de carga *point loads*, los cuales son iguales a los presentados en Escamilla, 1995.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
{0	0	0	1.307	0	0	0.645	−1.337	0	0.654	−2.317	0 }	^{t} 1×10^{-3} m

Tabla 1-9: Vector de desplazamientos de los nodos del modelo de la estructura para el patrón de carga *point loads*.

Vector de reacciones

Una vez se encuentran los desplazamientos en los nodos del modelo de la estructura, para cada uno de los patrones de carga, se procede a encontrar las fuerzas en los nodos producto de dichos desplazamientos.

En la tabla **1-10** se presenta el vector de fuerzas en los nodos del modelo de la estructura para el patrón de carga *point loads*, los cuales son iguales a los presentados en Escamilla, 1995.

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	
{-4	7	0	0	10	0	4	3	0	0	-20	0 }	^{t} t

Tabla 1-10: Vector de fuerzas de los nodos del modelo de la estructura para el patrón de carga *point loads*.

Procesamiento de los resultados

Hasta aquí, se ha solucionado el modelo de la estructura sometida al patrón de carga *point loads*. Con los resultados almacenados, se pueden determinar otros resultados que son interesantes. Para el caso concreto del modelo objeto de estudio en esta sección, se puede querer determinar las fuerzas internas de los elementos o las defomaciones en los mismos.

A modo de ejemplo, en la tabla **1-11** se presenta las fuerzas internas de cada uno de los elementos del modelo de la estructura sometidos al patrón de carga *point loads*, los cuales son prácticamente iguales a los presentados en Escamilla, 1995.

Elemento	Fuerza [t]
1-3	-11.667
1-4	13.333
3-2	-16.667
4-2	13.333
4-3	20

Tabla 1-11: Fuerzas internas de los elementos del modelo de la estructura para el patrón de carga *point loads*.

Bibliografía

Beer, F. (1997). *Mecánica vectorial para ingenieros*. México, McGraw-Hill.

Chacon, S. (2014). *Pro Git*. Berkeley, CA New York, NY, Apress,Distributed to the Book trade worldwide by Spring Science+Business Media.

Computers & Structures. (2017). *CSi Anlysis Reference Manual*.

Computers & Structures. (2019). *Welcome to ETABS*.

Computers & Structures. (2020). ETABS System Requirements [Accedido: 2020-09-29].

Escamilla, J. (1995). *Microcomputadores en ingeniería estructural*. Santafé de Bogotá, ECOE Universidad Nacional de Colombia. Facultad de Ingeniera.

Lutz, M. (2013). *Learning Python*. Sebastopol, CA, O'Reilly.

Wilson, E. L. & Dovey, H. H. (1972). Three dimensional analysis of building systems - TABS. *Earthwuake engineering research center*.

Wilson, E. L., Hollings, J. P. & Dovey, H. (1975). Three dimensional analysis of building systems (extended version). *Earthwuake engineering research center*.